Univerza v Ljubljani
Fakulteta za računalništvo in informatiko

Uroš Prosenik

Naravna interakcija za mobilno navidezno resničnost

DIPLOMSKO DELO
VISOKOŠOLSKI STROKOVNI ŠTUDIJSKI PROGRAM PRVE STOPNJE
RAČUNALNIŠTVO IN INFORMATIKA

izr. prof. dr. Iztok Lebar Bajec
MENTOR

Ljubljana, 2016

Tematika naloge:

Preučite aktualne razmere na področju interakcije z navidezno resničnostjo na mobilnih napravah. Zasnujte preprost algoritem, ki bo izkoriščal senzorski sistem mobilne naprave za doseganje čimbolj naravne interakcije uporabnika z navideznim okoljem.

IZJAVA O AVTORSTVU DIPLOMSKEGA DELA

Spodaj podpisani izjavljam, da sem avtor dela, da slednje ne vsebuje materiala, ki bi ga kdorkoli predhodno že objavil ali oddal v obravnavo za pridobitev naziva na univerzi ali drugem visokošolskem zavodu, razen v primerih kjer so navedeni viri.
S svojim podpisom zagotavljam, da:

- sem delo izdelal samostojno pod mentorstvom izr. prof. dr. Iztoka Lebarja Bajca,
- so elektronska oblika dela, naslov (slov., angl.), povzetek (slov., angl.) ter ključne besede (slov., angl.) identični s tiskano obliko in
- soglašam z javno objavo elektronske oblike dela v zbirki "Dela FRI".

— Uroš Prosenik, Ljubljana, september 2016.

Univerza v Ljubljani
Fakulteta za računalništvo in informatiko

Uroš Prosenik

Naravna interakcija za mobilno navidezno resničnost

POVZETEK

Navidezna resničnost (angl. virtual reality, VR) na tehnološkem trgu povzdiguje vse več prahu. Prav tako vse več mobilnih naprav, ki jih uporabniki uporabljajo za vsakodnevne potrebe, podpira in je dovolj zmogljivih za izvajanje VR aplikacij. Zaradi tega se na trgu pojavlja vse večje število VR očal, ki skrbijo za projekcijo slike z zaslona mobilne naprave na uporabnikove oči. Ta očala so lahko od različnih proizvajalcev in raznovrstnih oblik. Mnoga ne posedujejo nobenih dodatnih kontrolnih gumbov za interakcijo z mobilno napravo. Tako uporabnik, ko mobilno napravo enkrat vstavi v očala, ostane brez možnosti interakcije z navideznim svetom, saj nima dostopa do zaslona in gumbov naprave. Trenutno se ta problem v VR aplikacijah rešuje z osredotočenim pogledom uporabnika v objekt, s katerim želi izvesti interakcijo. Na primer uporabnik odpre vrata s pet sekund trajajočim pogledom usmerjenem na njih. Cilj diplomske naloge je zgoraj navedeno interakcijo nadgraditi in uporabnikom dodati možnost interakcije v navideznem svetu z naravnimi kretnjami. Namen je izdelati algoritem, ki bi skrbel za zaznavanje uporabnikovih interakcij v navideznem okolju iz uporabnikovih kretenj samo z uporabo senzorjev iz mobilne naprave. Ta bi tako nadomestil površine na dotik, upravljalnike na VR očalih ter vse ostale dodatne bluetooth upravljalnike, ki skrbijo za dodatne možnosti kontrol. Prav tako pa bo razvijalcem omogočena enostavna implementacija algoritma v njihove aplikacije. Tako se bodo odprle nove možnosti pri razvoju novih aplikacij in načinov iger.

Ključne besede: navidezna resničnost, interakcija v navideznem okolju

University of Ljubljana
Faculty of Computer and Information Science

Uroš Prosenik

Natural interaction for mobile virtual reality

ABSTRACT

Virtual reality (VR) has recently become a real hit. Also, an increasing number of mobile devices that are used for everyday needs support and are powerful enough to run VR applications. As a result, the market is growing in number of VR glasses, which project the image from mobile device screens to user eyes. These glasses can be from different manufacturers and different shapes. Many VR glasses do not provide any additional controllers for interaction with the mobile device. The user is limited, once the mobile device is inserted into the eyewear, they remain without any means to interact with the virtual world, because there is no way to access the screen and buttons of the device. Currently, these problems in VR applications are solved with the use of the user's gaze direction. For example, the user opens the door by staring at them for 5 seconds. The goal of this thesis is to develop new interaction techniques and give users the ability to interact with a virtual world with natural body movements. The task is to create an algorithm which would be responsible for detecting interaction from the user's gestures using only sensors already built in mobile devices. This would replace touch pads and buttons on VR glasses and all other additional Bluetooth controllers that provide additional control options. Also, it will allow developers to easily implement this algorithm in their applications and give them more options for the development of new applications and ways of gaming.

Key words: virtual reality, interaction in virtual enviroment

ZAHVALA

Zahvaljujem se mentorju izr. prof. dr. Iztoku Lebarju Bajcu za strokovno pomoč, usmerjanje in vse podane nasvete skozi celotno izvedbo doplomskega dela. Zahvalil bi se tudi družini, Tjaši in vsem testirancem.

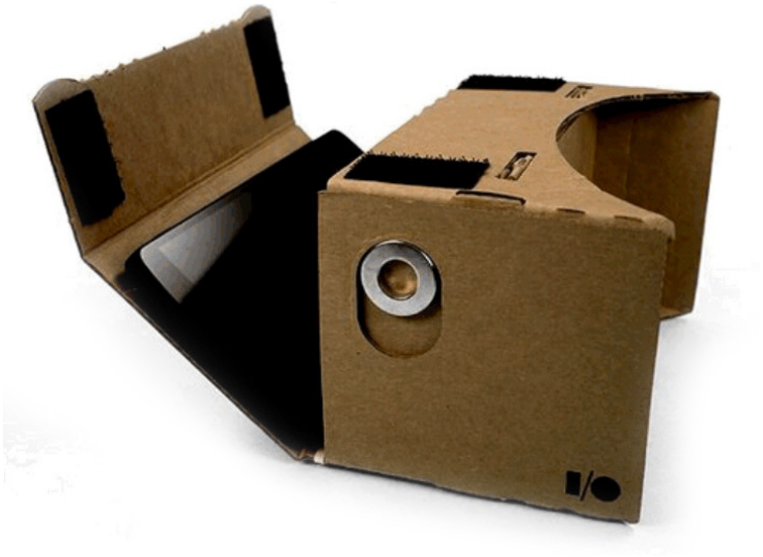
— Uroš Prosenik, Ljubljana, september 2016.

KAZALO

1 Uvod

Navidezna resničnost (angl. virtual reality, VR) je izraz, ki opisuje tridimenzionalno računalniško generirano okolje, katerega je mogoče raziskovati. Oseba z uporabo različnih VR naprav postane del tega okolja in v njem lahko izvaja različne interakcije, medtem ko se navidezno okolje na te interakcije uporabniku ustrezno odziva. Tehnologije za izvedbo in prikaz navidezne resničnosti poznamo že kar nekaj časa. Njeni začetki segajo v sredino prejšnjega stoletja. Tehnologija se je od takrat počasi razvijala, njen razvoj pa je dosegel vrhunec šele v zadnjem desetletju. Za to je deloma poskrbelo tudi podjetje Oculus z izdelkom Rift.[1] Kmalu po njegovem izidu, so zaradi silovite uspešnosti in zanimanja po tovrstnem izdelku, svoje VR naprave začela razvijati tudi druga podjetja kot so Google, Htc, Sony, Playstation in drugi. Danes mnoga podjetja tekmujejo, katero bo izumilo bolj inovativno napravo in prevzelo VR trg. Te naprave so trenutno še vseeno nekoliko drage in za izvajanje aplikacije in izris slik na zaslone v ozadju potrebujejo zmogljive in posledično drage računalnike. Prav zaradi tega so mnogim uporabnikom še vedno nedostopne. Ker pa vse več uporabnikov poseduje vse bolj zmogljive mobilne naprave,

[1] https://www3.oculus.com/en-us/rift/



Slika 1.1 Magnetni pomični gumb na Google cardboard očalih.

so VR tehnologijo razširili tudi na mobilne naprave. Ta je cenovno bistveno bolj dostopna, saj poleg mobilne naprave, ki podpira VR, uporabnik potrebuje le dodatna očala za prikaz VR aplikacij. Ta skrbijo, da se slika iz mobilne naprave pravilno prikaže uporabniku na oči. Seveda so te VR aplikacije precej bolj enostavne kot računalniške VR aplikacije, vseeno pa nudijo uporabniku precej dobro predstavitev simulacije navideznih svetov. Ko je mobilna naprava nameščena v očala, uporabnik v velikih primerih nima dostopa do kakršne koli interakcije z mobilno napravo, saj je ta zaprta znotraj očal. Posledično ostane brez možnosti interakcije tudi v navideznem okolju. Mnogi proizvajalci ta problem rešujejo z implementacijo magnetnega gumba na VR očalih. Ta po navadi leži na eni izmed strani očal, uporabniku pa nudi možnost klik-interakcije na mobilno napravo (glej sliko 1.1).

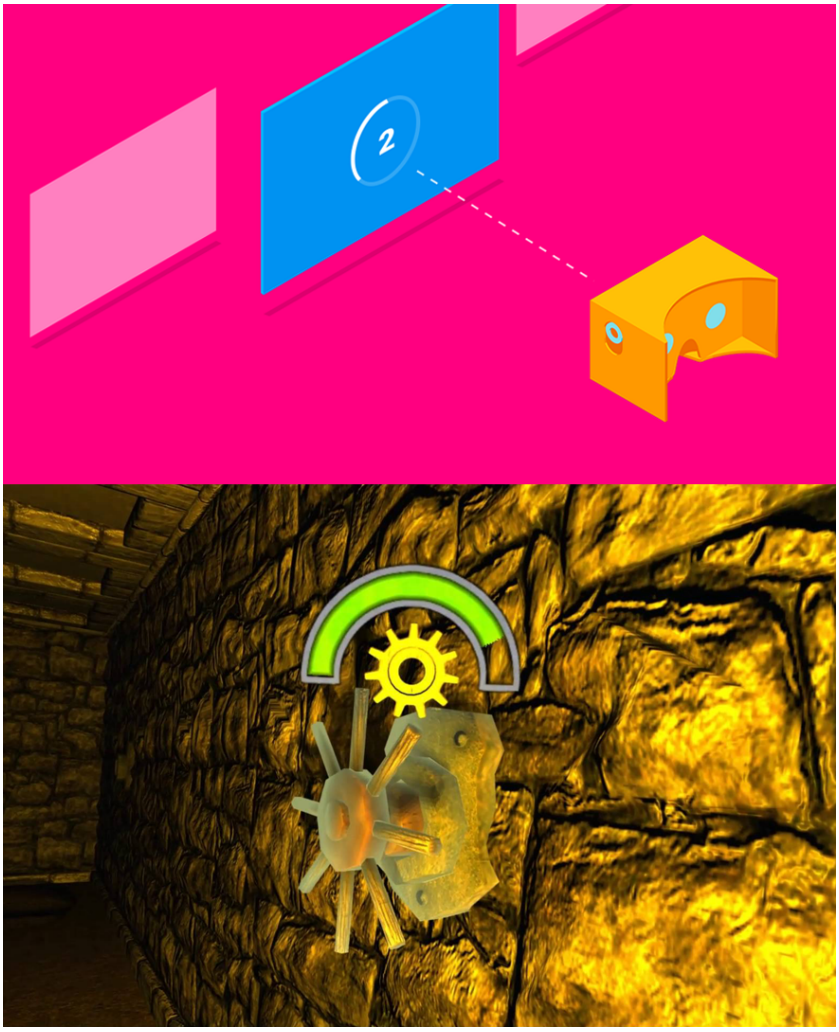
Dodatne možnosti ponujajo priloženi bluetooth upravljalniki. Ti se z mobilno napravo povežejo preko povezave bluetooth in uporabniku nudijo različne variacije kontrol, te pa so odvisne od vsakega upravljalnika posebej. V večini primerov so to preprosti kontrolniki z eno kontrolno palico in od 2 do 6 gumbov (glej sliko 1.2).

Mnoga očala pa nimajo magnetnih gumbov in pridejo v paketu brez kakršnega koli bluetooth upravljalnika. Tako ostane uporabnik, ko svojo mobilno napravo vstavi v očala, v navideznem okolju brez dodatnih upravljalnih možnosti. Edini način interakcije,

Slika 1.2 Bluetooth upravljalnik.

je uporaba smeri pogleda, ki ga naprava zaznava z vgrajenimi senzorji (glej sliko 1.3). Uporabnik z obračanjem glave usmeri pogled v igri na objekt, v katerega mora biti določen čas zazrt. Po določenem pretečenem času se interakcija nad objektom zazna in izvede.

Cilj diplomske naloge je rešitev zgoraj navedenega problema. Končana spisana knjižnica v okolju Unity bo uporabnikom podala nove možnosti interakcije v navideznem okolju brez potrebe po zapravljanju denarja za dodatne upravljalnike. Knjižnica bo namesto zunanjih upravljalnikov z uporabo senzorjev v mobilni napravi prepoznavala gibe uporabnika in skrbela za aktivacijo kontrol v igri. Tako bodo mnogi razvijalci lahko v svoje projekte vpeljali nove načine kontrol, ki jih prej morda ni bilo mogoče zaradi nedosegljivosti kotrol mobilne naprave zaprte v ohišju VR očal.



Slika 1.3 Zgornji del slike je shematski prikaz interakcije z usmerjenim pogledom, spodnji del pa implementacija le-te v igro. Uporabniku je potrebno podati vizualno povratno informacijo, koliko časa mora še vztrajati s pogledom, da se bo interakcija uspešno izvedla.

2 Zgodovina

Začetki razvoja navidezne resničnosti segajo vse do sredine prejšnjega stoletja [1]. Prvo zabeleženo znanstveno-fantastično delo, ki raziskuje navidezno resničnost, je nastalo leta 1935. Delo, poimenovano *Pygmalion's Spectacles*, je napisal Stanley Grauman Weinbaum.[1] V njem opisuje sistem, ki z vizualnimi učinki, zvokom, dotikom in celo vonjem simulira navidezno okolje.

Morton Heilig je leta 1950 napisal delo *Experience Theatre*, katerega prototip poimenovan Sensorama je izdelal leta 1962. Stroj je predvajal pet kratkih filmov, ob tem pa je s premikanjem stola in spuščanjem različnih vonjav simuliral občutek bivanja v navideznem okolju[2].

Leta 1966 se je navidezna resničnost razširila iz zabavne industrije v vojaško. Thomas A. Furness je ameriški zračni sili predstavil navidezno resničnostni simulator letenja in tako razširil uporabo navideznih okolij (glej sliko 2.1). S tem izdelkom si je prislužil vzdevek stari oče navidezne resničnosti.[2]

V tistem času pa so izumili tudi prvi koncept obogatene resničnosti (angl. augmented

[1] https://en.wikipedia.org/wiki/Stanley_G._Weinbaum
[2] https://en.wikipedia.org/wiki/Thomas_A._Furness_III

Slika 2.1 Simulator letenja predstavljen leta 1966.

reality, AR).[3] Danes produkt Hololense podjetja Microsoft[4] za prikaz navideznega okolja uporablja podobno tehnologijo. Naprava z uporabo dveh katodnih cevi na vsako oko projicira računalniško generirane slike. Uporabnik tako vidi 3D slike locirane na resničnih objektih v okolici. Prvi prototip naprave je bil tako neroden in težek, da ga je bilo potrebno pritrditi na premično roko, ki je visela s stropa. Napravo prikazuje Slika 2.2.

Od tu naprej pa se je razvoj VR tehnologije upočasnil in nekoliko zapadel v ozadje. Tehnologija in naprave razvite v tem času seveda predstavljajo napredek v razvoju, vendar so bile prevelike, nerodne in precej drage. Razvite so bile različne tehnike prikazovanja navidezne resničnosti, različni VR upravljalniki in nove naprave za prikazovanje navideznih okolij. Nekatere od tehnologij razvitih v tistem času se deloma uporabljajo še danes. Tako je navidezna resničnost nekoliko potihnila v ozadje vse do razvoja naprave Oculus Rift.[5] Oculus je bil predstavljen na Kickstarterju leta 2012, kjer so zanj zbrali neverjetnih 2,5 milijona ameriških dolarjev. Od tu naprej pa sta zanimanje in razvoj VR

[3] https://en.wikipedia.org/wiki/Augmented_reality
[4] https://www.microsoft.com/microsoft-hololens/en-us
[5] https://www3.oculus.com/en-us/rift/

Slika 2.2 Prvi prikaz naprave, ki omogoča izvedbo obogatene resničnosti.

razcvetela. Trenutno najbolj poznane naprave za prikaz VR okolij so Oculus Rift, Sony Morpheus,[6] Fove,[7] Razer OSVR[8] in HTC Vive.[9] Te pa so še vseeno nekoliko drage in nedostopne vsem uporabnikom.

Vse več uporabnikov za vsakodnevne potrebe uporablja mobilne naprave, ki so dovolj zmogljive za predvajanje enostavnih VR aplikacij. Zaradi tega se je navidezna resničnost preselila tudi na mobilne naprave. Uporabniki tako lahko s poceni opremo tudi sami doživijo kanček navidezne resničnosti. Prav tako je enostavna prenosljivost, saj uporabnik poleg mobilne naprave za prikaz navideznega okolja potrebuje le še dodatna VR očala, ki pravilno preslikajo sliko z mobilne naprave na oči uporabnika. Samsung Gear VR,[10] Google Cardboard[11] ter mnogo drugih ponaredkov se že uspešno prodaja na trgu. Vse več se na trgu pojavlja tudi aplikacij, ki so namenjene za mobilni VR. Te so dosti bolj enostavne kot tiste za osebni računalnik ali konzole, saj mobilne naprave še vseeno niso toliko zmogljive in ne zmorejo predvajati tako zahtevnih aplikacij kot recimo osebni

[6] https://www.playstation.com/en-us/explore/playstation-vr/
[7] http://www.getfove.com/
[8] http://www.razerzone.com/osvr
[9] https://www.htcvive.com/eu/
[10] http://www.samsung.com/global/galaxy/gear-vr/
[11] https://vr.google.com/cardboard/

računalnik. Obstajajo pa tudi aplikacije za mobilne naprave, ki z uporabo brezžične povezave oziroma USB povezave, na mobilni napravi prikazujejo sliko, prejeto z računalnika, medtem ko se VR aplikacija poganja na njem. Primer takšne aplikacije je Trinus VR.[12] Tako lahko z uporabo mobilne naprave in VR očal dobimo približek dragih VR naprav za prikazovanje navideznih okolij, saj lahko poganjamo ne samo mobilne, ampak tudi računalniške VR aplikacije.

[12] http://trinusvr.com/

3 Vizualizacija navideznih okolij

Zavajanje oči s prikazovanjem navideznih okolij z vnaprej prirejenimi slikami so poznali že v začetku 50. let prejšnjega stoletja. Tehnike današnjih naprav se od starih razlikujejo v zaznavanju gibanja in njegovi uporabi pri izračunavanju in posodabljanju izrisa slike. Medtem, ko so včasih slike vnaprej ustrezno pripravili, danes s prejetimi podatki o gibanju iz VR naprave računalnik preračunava in spreminja prikaz slike okolja. To tehniko uporablja večina današnjih VR naprav. Naprave uporabniku skozi leče projicirajo sliko iz dveh oziroma enega razdeljenega zaslona na oči. Za zaznavanje gibanja večina naprav uporablja različne senzorje, kot so na primer merilec pospeška, žiroskop, magnetometer in kamera. Naprava vse te podatke preko kablov pošilja v računalnik. Aplikacija to informacijo uporabi, preračuna in vrne napravi podatek v obliki ustrezno spremenjene slike, ki jo izriše na zaslon. Tako uporabnik dobi občutek obračanja pogleda v navideznem okolju. Primer spreminjanja slike je, ko uporabnik zavrti glavo, naprava to zazna, podatke pošlje aplikaciji, ta pa z njimi sliko preračuna in predstavi v obratni smeri vrtenja glave. Seveda pa mora biti hitrost vrtenja slike ustrezno prilagojena hitrosti vrtenja glave, da predstavitev izpade resnična. Slika 3.1 prikazuje notranjost ene od naprav za

Slika 3.1 Razstavljena VR naprava OSVR podjetja Razer.

prikaz navidezne resničnosti.

Ker pa vse tehnike temeljijo na zavajanju človeških čutil, je potrebno vse skupaj predstaviti čim bolj natančno, brez zakasnitev in s čim hitrejšim osveževanjem ekrana. Potrebnih je vsaj 60, priporočenih pa vsaj 90 slik na sekundo. Če tehnika prikazovanja ni popolna, lahko pride do pojava gibalne slabosti (angl. motion sickness), zaradi katerega oseba postane omotična in se ji lahko pojavi tudi slabost. Ta predstavlja velik problem pri razvoju navideznih okolij, še vedno pa si strokovnjaki niso enotni, kaj točno jo povzroči [3].

Naprava FOVE ima poleg vseh senzorjev, ki skrbijo za zaznavo premikov, razvito sledenje oči. Za to poskrbijo infrardeče kamere, ki sledijo zenicam v očeh. S to tehnologijo so pridobili dodaten način interakcije z okoljem in njegovim prikazom. Medtem, ko obračanje glave skrbi za spreminjanje prikaza slike, oči poskrbijo za smer pogleda v navideznem okolju. Ta tehnologija omogoča nove možnosti, kot je globinska ostrina slike v smeri [4]. To pripomore pri ublažitvi napora oči, posledično pa zmanjša možnost pojavitve gibalne slabosti [5].

3.1 VR očala

Najbolj enostaven in najcenejši način predstavitve navideznih okolij je z uporabo mobilnih naprav. Sam način prikazovanja je podoben kot pri dosedaj omenjenih napravah, le da se za zaznavanje gibov uporabljata merilec pospeška in žiroskop, deloma lahko tudi kamera. Ti senzorji sicer zadoščajo za predstavitev VR, a pri velikem številu mobilnih naprav še vedno niso dovolj natančni. Prav tako je problem prikazovati zahtevne VR aplikacije, saj mobilne naprave niso dovolj zmogljive. Zahtevna aplikacija posledično deluje počasi, ne dosega prikaza 60 slik na sekundo in se ne odziva kot želeno. Na spletu pa je vse več VR aplikacij prilagojenih za širok spekter mobilnih naprav, ki so trenutno prisotne na trgu. Aplikacije so manj zahtevne, imajo manjše teksture in objekte z majhnimi števili poligonov. Mobilne naprave tako lažje dosegajo boljše rezultate. Z uporabo posebnih očal se slika aplikacije preslika z ekrana mobilne naprave. Na njenem zaslonu se slika razpolovi, leči VR očal pa vsaka na svoje oko prikazujeta svoj del slike. Vsa VR očala uporabljajo isto tehniko preslikave, in sicer z uporabo dveh ukrivljenih leč. Seveda pa se kvaliteta slike in udobje naprav med seboj razlikujejo.

3.2 Projekcija

Navidezna resničnost je lahko predstavljena tudi s projekcijo. Sistema CAVE in Cybersphere uporabljata projekcijo slike navideznih okolij ter s tem uporabnika prepričata, da se nahaja v navideznem okolju [6]:

- CAVE je sistem, ki projicira sliko v prostoru kockaste oblike na vse zidove. Prostor lahko hkrati obišče več oseb. Te si nadenejo 3D očala, ena od oseba pa uporablja dodatno napravo, ki zaznava gibe glave. Tako se ves čas perspektiva projiciranih slik preračunava in spreminja glede na smer pogleda uporabnika z dodatno napravo [7].

- Cybersphere, prav tako kot CAVE, za prikaz navidezne resničnosti uporablja projekcijo slike na steno prostora. Pri Cybersphere je ta prostor velika krogla, v kateri je uporabnik, slika pa se projicira na zunanje strani krogle. Ta stoji na stojalu z valjčki, ki omogočajo vrtenje krogle v vse smeri. Valjčki vsebujejo senzorje, ki zaznavajo smer vrtenja krogle. Ta podatek se uporablja za posodabljanje slike, ki je projicirana na kroglo. Uporabnik tako za zaznavo gibanja ne potrebuje nobenih

Slika 3.2 Crybersphere v uporabi.

dodatnih naprav. Slika 3.2 prikazuje uporabnika v krogli. Za premikanje v navideznem okolju uporabnik enostavno hodi po krogli in jo tako vrti v željeno smer gibanja. Ob vrtenju krogle senzorji v valjčkih podatke pošiljajo računalniku, ta pa podatke uporabi, da pravilno spremeni sliko, ki se projicira na kroglo.

4 Interakcija v navideznih okoljih

Človek pri interakciji z resničnim svetom od okolja pridobiva dotični in vizualni odziv. Prav tu pride do glavnih razlik med realnim in navideznim okoljem. Trike za zavajanje oči poznamo že dolgo. Tudi navidezno resničnost so kaj kmalu znali predstaviti uporabniku. Problematično je le, ko ta želi z njo vzpostaviti stik, oziroma od nje pridobiti dotični odziv. Velik poudarek razvoja se namenja tej smeri tehnologije. Naprave, ki prikazujejo navidezno resničnost, lahko za dodatno interakcijo z okoljem, poleg vseh senzorjev, uporabljajo dodatne upravljalnike. Različne naprave podpirajo različne upravljalnike in so nanje tehnično prilagojene. Nekatere od teh za podajanje čutnih odzivov uporabljajo vgrajene komponente, ki vibrirajo. Ostale naprave pa ne vsebujejo nobenih komponent za podajanje čutnih odzivov. Tako poznamo več načinov interakcije v navideznih okoljih.

4.1 Interakcija brez upravljalnikov

Veliko število VR aplikacij rešuje problem, kot je pomanjkanje upravljalnih kontrol za interakcijo, s tako imenovanim konceptom usmeritve pogleda za uspeh (angl. stare to win) [8]. Gre za interakcijo, ki temelji na določen čas trajajočem pogledu usmerjenem

na objekt, nad katerim se interakcija želi izvajati. Recimo, da želi uporabnik odpreti vrata. To stori tako, da usmeri pogled proti vratom in je vanje usmerjen toliko časa, kolikor je določen interakcijski čas v aplikaciji. Ko se igralec odloči, da želi nad objektom izvesti interakcijo, mora tako kljub odločitvi še vedno počakati določen čas, da računalnik njegovo želeno interakcijo zazna, kar pa ni v redu. Uporabnik se je že odločil kaj želi storiti in pričakuje od aplikacije, da se na njegove zahteve odziva v istem trenutku, kot so te nastale. To lahko povzroči občutek neodzivnosti aplikacije [9].

4.2 Upravljalniki s prostorskim zaznavanjem

Upravljalniki s prostorskim zaznavanjem vsebujejo podobne senzorje kot sama naprava za prikaz navidezne resničnosti, merilec pospeška in žiroskop. Poleg senzorjev pa imajo še dodatne gumbe. Upravljalniki služijo kot dodatna možnost interakcije v navideznem okolju in se jih lahko tudi ustrezno, glede na aplikacijo v navideznem okolju grafično prikaže. Upravljalnik ves čas zaznava gibe, podane ukaze, rotacije in lokacijo v prostoru [10]. Od tu naprej pa vse upravlja aplikacija sama: kakšen bo prikaz upravljalnikov, kako se bo z njimi upravljalo in kakšne bodo interakcije. Za vse to poskrbi razvijalec aplikacije. Primer takih upravljalnikov uporabljata naslednji podjetji:

- Oculus z izdelkom Rift[1] uporablja dva, tako imenovana Touch upravljalnika.[2] Upravljalnik je ročajaste oblike z ovalnim obročem okoli dlani. Ta uporabniku pomaga pri določitvi središča upravljalnika, torej centra vseh rotacij. Na napravi se nahaja tudi 5 gumbov in kontrolna palica (glej Slika 4.1).
- HTC Vive[3] upravljalnika sta podolgovate oblike. Poleg vseh prostorskih senzorjev ima 5 gumbov in površino občutljivo na dotik (angl. touch pad). Ta razvijalcem ponuja širok spekter možnosti implementacije različnih načinov kontrol, saj se jo da poljubno reprogramirati. Upravljalnika prikazuje slika 4.2.

4.3 Upravljalniki na osnovi kamere

Upravljalniki na osnovi kamere namesto fizičnih gumbov in upravljalnikovih premikov, snemajo in zaznavajo gibe uporabnika. Njegovim premikom kamera sledi in pridobljene

[1] https://www3.oculus.com/en-us/rift/
[2] https://www3.oculus.com/en-us/touch/
[3] https://www.htcvive.com/eu/

Slika 4.1 Touch upravljalniki podjetja Oculus.

Slika 4.2 Vive upravljalniki podjetja HTC.

Slika 4.3 Zaznavanje in izdelava okostja z uporabo Leap Motion kamere.

podatke pošilja v računalnik oziroma konzolo. V večini primerov kamera z uporabo različnih algoritmov, po obliki dela telesa, ki ga snema, ustvari ustrezno oblikovano okostje. To okostje sledi gibom telesa in ustrezno spreminja svoje pozicije in rotacije sklepov. Premike kamera pretvori v vrednosti in jih pošlje v računalnik oziroma konzolo, kjer se nato v aplikaciji ustrezno pretvorijo v interakcijo. Omenimo dva taka produkta:

- Leap Motion je kamera, ki se jo namesti pred uporabnika na mizo. Kamera zaznava obliko dlani. Ko jih uporabnik postavi nad kamero, ta z uporabo algoritma sestavi okostje v obliki dlani (glej sliko 4.3). Okostje sledi gibom in rotacijam uporabnikove dlani, to pretvori v vrednosti in jih pošilja v računalnik. Tako lahko uporabnik v navideznem okolju izvaja interakcije z uporabo lastnih rok. Kako pa ti premiki izvajajo interakcijo v navideznem okolju pa je odvisno od aplikacije same in različnih vnaprej sprogramiranih algoritmov zaznavanja. Leap Motion ima postavljeno tudi stran za razvijalce, na kateri se nahaja mnogo primerov skript, forumov, razprav ter knjižnic drugih razvijalcev.[4]
- VR Developer Mount je podobna izvedba s podobnim načinom uporabe kot Leap

[4] https://developer.leapmotion.com/

Motion, le da se ta namesto na mizo pritrdi na navidezna očala. Naprava je namenjena zaznavanju gibov, ki jih uporabnik izvaja z rokami pred seboj v smeri pogleda. Ta način poskrbi za še boljši približek resničnosti, saj se tako lahko uporabnik prosto premika po prostoru in njegov gibalni prostor ni omejen le na mizo.

4.4 Kombinacija obeh

Upravljalniki v tej skupini uporabljajo tehnologijo prostorskega zaznavanja in zaznavanja s kamero. V večini primerov kamera zaznava del očal oziroma upravljalnika, ki sta primerno osvetljena oziroma označena ter s tem locira njuno lokacijo in premike v prostoru. Senzorji v napravi za prostorsko zaznavanje skrbijo za rotacije in pomoč kameri pri določitvi njihove lokacije, premikov in rotacij.

Primer takega tipa upravljalnika je Playstaton Move (glej sliko 4.4), ki za pravilno delovanje potrebujejo dodatno kamero. Ta s pomočjo svetlobe zaznava lokacijo komponent in njihovo premikanje. Tako kot Playstation VR naprava za prikaz navideznega okolja, tudi priloženi upravljalniki vsebujejo svetilne komponente, ki jim kamera lahko sledi. Na vrhu upravljalne palice se nahaja barvna lučka, kateri lahko kamera, zaradi velike vizualne razlike z okolico, med gibi lažje sledi. Glede na njeno zaznano lokacijo si preračunava podatke o prostorskem gibanju in lokaciji.

Slika 4.4 Move upravljalnik podjetja Playstation s kamero.

5 Človeška čutila in kretnje

Čutila, s katerimi prepoznavamo okolico, v kateri se nahajamo, so vid, sluh, vonj in dotik. Če nam vsa ta čutila pošiljajo ustrezne dražljaje, nam to daje občutek resničnosti in obstojnosti okolja okoli nas [11]. Kako pa simulirati navidezni prostor s tehnologijo VR in s tem prelisičiti čutila v obstoj navideznega okolja okoli nas?

Eno izmed glavnih čutil za dojemanje okolice je vid. S premikanjem glave spreminjamo sliko, katero prikazuje VR naprava. Usmerjenost oči pa nam, v primeru da VR naprava to zaznava, ustrezno fokusira vidno sliko in njeno globinsko ostrino. Usmerjenost oči je tehnično nekoliko težje zaznavati, vendar produkt FOVE[1] to že uspešno izvaja. Poleg gibov glave zaznavajo tudi gibe oči. Ostale naprave, trenutno na trgu, njihovega premikanja še ne zaznavajo [12].

Večina ostalih naprav za prikaz navideznih okolij zaznava gibe glave [13]. S premikanjem in obračanjem glave vgrajeni senzorji kot so merilec pospeška, žiroskop, magnetometer in kamera, zaznavajo premike. Te podatke ustrezno pretvorijo in jih uporabijo za premik slike na zaslonu naprave. Tako uporabnik dobi občutek obračanja v navide-

[1] http://www.getfove.com/

znem okolju [14]. Za zaznavanje teh gibov se v večini uporabljata merilec pospeška in žiroskop. Senzorja zaznavata tudi premike po prostoru, vendar se tu pojavi problem. Če navidezno okolje ni predstavljeno enako kot prostor, v katerem se uporabnik nahaja, ta zaradi nadetih VR očal ne vidi prostora okoli sebe in je primoran se slepo gibati po prostoru. To pa je lahko nevarno. Zaradi tega se ta tehnika uporablja le v primeru, ko je prostor vnaprej pravilno pripravljen glede na prostor v VR aplikaciji. Prav tako so pri gibanju težava kabli. Problem rešujejo izdelki, kot so Virtualizer podjetja Cyberith[2] in Omni podjetja Virtuix.[3] To sta prilagojeni podlagi, po kateri uporabnik med gibanjem drsi, pri tem pa ostaja ves čas na istem mestu. Podlaga zaznava smer in hitrost gibanja ter podatke posreduje aplikaciji. Uporabnik je okoli pasu pripet na nosilec za potrebno upornost pri drsenju, istočasno pa ta meri višino pasu v prostoru. Med igro lahko iz teh parametrov računalnik ugotovi, ali uporabnik čepi, stoji ali skače.

Za prepoznavanje človeških kretenj sem uporabil merilec pospeška. Z uporabo dodatnih senzorjev, kot sta žiroskop in GPS bi program sicer pridobil natančnost pri zaznavi in možnost implementacije dodatnih kretenj, a dodatnih senzorjev nisem vključil, saj jih mnoge mobilne naprave nimajo in bi s tem omejil ciljno populacijo.

Merilec pospeška, kot že njegovo ime pove je senzor, ki pridobiva podatke o pospeških. Je senzor, katerega ima trenutno večina mobilnih naprav na trgu. Njegova glavna vloga na mobilnih napravah je prilagajanje orientacije slike na zaslonu, glede na to kako je mobilna naprava pozicionirana. Uporablja se tudi za zaznavo premikov mobilne naprave in za kontrole različnih mobilnih igric.

Slika 5.1 prikazuje smeri koordinatnih osi merilca pospeška in žiroskopa, ki so enake tako v vertikalni kot v horizontalni poziciji mobilne naprave.

5.1 Zaznavanje hoje na mestu

Sile, ustvarjene med hojo, se prenesejo na VR napravo, ki je nameščena na uporabnikovi glavi. S ciljem zaznavanja hoje morajo biti sile natančno izmerjene. To predstavlja težavo pri napravah z nekoliko slabšimi merilnimi senzorji. Kot omenjeno se bodo meritve hoje izvajale nad kretnjami izvedenimi na mestu. Horizontalne sile tako lahko deloma zanemarimo in se bolj posvetimo vertikalnim silam.

Seveda postanejo meritve bolj natančne, če se algoritmu v zaznavanje dodajo poleg

[2] http://cyberith.com/product/
[3] http://www.virtuix.com/

vertikalnih sil tudi horizontale, a se je po testiranjih izkazalo, da aplikacija deluje bolj odzivno brez njih. Zaznavanje horizontalnih sil sem tako v algoritmu izpustil. Z uporabo samo vertikalnih sil, je bilo iz testiranj razvidno, da začetna kalibracija mejnih vrednosti zaznave ni bila potrebna, saj so bili podatki vseh testirancev dovolj podobni in je algoritem uspešno zaznaval hojo pri vseh testirancih, ko so se ti nanj privadili.

5.2 Zaznavanje skoka na mestu

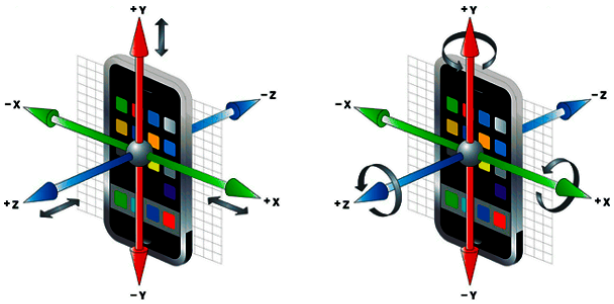
Zelo pomemben gib v industriji iger je skok. Ko človek skoči, v kratkem času zelo poskoči pospešek na vertikalni osi. Prav tako je potrebno za odzivnost igre iz senzorjev dovolj hitro prepoznati gib, da ne pride do prevelikih zakasnitev.

V začetni fazi oseba nekoliko zaniha proti tlom. To predstavlja odriv, pri katerem se telo nekoliko približa k tlom, da se pripravi na odskok. Po tem vertikalni pospešek zelo hitro poskoči, kar predstavlja odskok. Pri pristanku pa sila drastično upade glede na sile v začetku, kar predstavljata pristanek in upogib nog ob pristanku. Te podatke iz merilca pospeška ustrezno podajajo vertikalnie sile. Tu nas horizontalni podatki izmerjenih sil ne zanimajo, saj nimajo nikakršnega vpliva pri zaznavi skoka na mestu. Horizontalne sile v smeri naprej-nazaj bi lahko uporabili za zaznavanje skoka naprej in nazaj, medtem ko bi sile v smeri levo-desno uporabili za skok v levo oziroma desno. Kot omenjeno, je potrebno paziti pri času potrebnem za prepoznavanje zajetih podatkov, da v aplikaciji ne pride do prevelikih zakasnitev.

Sile izvedene pri skoku so dovolj specifične, da gib skoka algoritem prepozna že samo iz njegovih prvih dveh faz: iz rahlega počepa ter odskoka. V uspešnih pogojih zaznava kretnje traja 2 do 5 časovnih korakov definiranih v algoritmu. Vse ostale sile, ki nastanejo pri skoku po teh dveh kretnjah lahko zanemarimo. S tem sicer nekoliko izgubimo natančnost zaznave skoka, veliko pa pridobimo na hitrosti zaznave skoka in s tem dosežemo boljšo odzivnost v aplikaciji.

5.3 Zaznavanje meta na mestu

Za met predmeta so uporabljeni podatki o rotaciji po vertikalni osi. Ko človek vrže predmet, se zgornji del njegovega trupa v zelo kratkem času nekoliko rotira v smeri meta in nato povrne v prejšnje stanje. Te sile se deloma prenesejo tudi na glavo in posledično na VR napravo, ki je nameščena na njej. Tu pa lahko pride do napačnih ugotovitv v



Slika 5.1 Smeri koordinat merilca pospeška (levo) in žiroskopa (desno) v mobilnih napravah. Smeri koordinat so iste v horizontalni in vertikalni poziciji mobilne naprave.

algoritmu, saj je lahko hiter obrat uporabnika v horizontalni ravnini zaznan kot met. Zaradi tega je potrebno izmerjenim silam pri metu natančneje določiti mejne zaznavne vrednosti in dodati zaznavanje sil tudi po vertikalni osi [15, 16].

Za pravilno zaznavanje meta je začetna kalibracija uporabnika obvezna. Medtem ko so sile pri hoji in skoku podobne pri večini uporabnikov, kar je bilo prav tako razvidno iz podatkov pridobljenih pri meritvah, se te zelo razlikujejo pri metu. Ta problem v večini rešuje začetna kalibracija, ki natančneje določi meje in logiko privzetega algoritma za zaznavanje meta vsakega uporabnika posebej.

5.4 Zaznavanje ostalih kretenj

Težava pri uporabi merilca pospeška mobilne naprave je, da lahko pride v algoritmu do napačnih zaznav pri kretenjah, ki se ne izvedejo v kratkem časovnem obdobju. To oteži implementacijo drugih interakcij, kot je na primer počep. Z lahkoto bi algoritem zaznal, kdaj uporabnik počepne in kasneje tudi vstane, če bi ta to storil vedno na enak način in z enako intenziteto. Do napak bi prišlo v primeru, ko bi uporabnik prepočasi počepnil oziroma vstal, da bi merilec pospeška to zaznal. Zgodilo bi se lahko, da bi uporabnik ustrezno počepnil in kasneje premalo sunkovito vstal. Tako bi algoritem počep zaznal, za povratek pa bi bile sile pod mejami zaznave algoritma. Posledično bi stanje igralca v igri ostalo v počepu. Takšni pojavi bi pokvarili uporabniško izkušnjo in onemogočali zaznavo naslednje izvedene kretnje. Slednje bi deloma lahko rešili z uporabo vgrajenega GPS senzorja v mobilni napravi, a je ta pri mnogih napravah premalo natančen in ima preveč občasnih napak in poskokov v meritvah.

6 Izdelava aplikacije

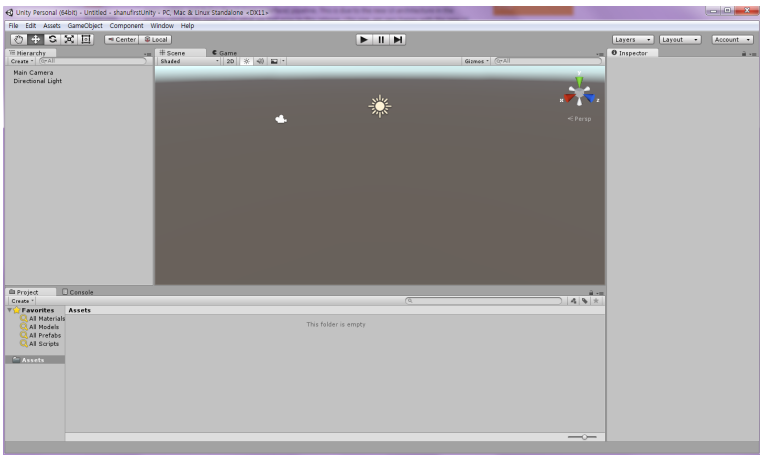
Za testiranja in celoten razvoj aplikacije sem uporabil okolje Unity 3D.[1] To okolje je namenjeno predvsem razvoju iger ter spletnih aplikacij. Ima prijazen in preprost uporabniški vmesnik (glej sliko 6.1). Podporo nudi različnim platformam kot so Windows, Mac, Linux, iOS, Android, Playstation, Xbox 360, Nintendo Wii, WebGL in spletni vtičnik za brskalnike imenovan Unity Web Player. Prav tako ima okolje že vgrajeno podporo za fiziko. Za fiziko v ozadju skrbi fizikalni pogon PhysX, ki ga je razvilo podjetje nVidia.[2]

Okolje uporablja tri grafične knižnice:

- Direct 3D za platformi Windows in Xbox 360,
- OpenGL za platformi Windows in Mac,
- OpenGL ES za Android ter iOS naprave.

VR aplikacija, ki sem jo razvil v sklopu diplomske tako uporablja knjižnico OpenGL ES za Android.

[1] http://en.wikipedia.org/wiki/Unity_(game_engine)
[2] http://www.nvidia.com/



Slika 6.1 Unity 3D - Glavno okno.

Velika prednost okolja Unity 3D je sposobnost prenosljivosti iste aplikacije na več platform. Programski jeziki, ki jih okolje podpira so C#, prilagojeni Java Script in programski jezik Boo. Okolje Unity 3D ima tudi dobro dokumentacijo. Lastijo nekaj razvijalcem namenjenih internetnih strani, kjer uredniki in razvijalci okolja Unity 3D skupaj z ostalimi uporabniki strani odgovarjajo na zastavljena vprašanja in rešujejo težave uporabnikov. Prav tako ima okolje svojo trgovino s sredstvi (angl. Asset Store)[3], v kateri se nahajajo programske kode, že vnaprej narejene scene, animacije, knjižnice, objekti in še mnogo drugih stvari. Tu si razvijalec lahko kupi oziroma pridobi zastonj katerokoli od obstoječih sredstev in jih vključi v svoj projekt. Ta trgovina razvijalcem prihrani veliko časa in omogoči tistim razvijalcem, ki imajo pomanjkanje znanja na določenem področju do uspešne dopolnitve njihovega projekta. Prav tako v programu, če to ni posebej zahtevano, ni potrebno navajati nobenih pravic od uporabljenih sredstev.

6.1 3D modeli, animacije in okolje aplikacije

Modeli, ki sestavljajo sceno razvite VR igre so pridobljeni v trgovini s sredstvi okolja Unity 3D. Modeli (glej sliko 6.2) so bili brezplačni in preprosti. Paziti sem moral, da jih je v igri čim manj in da so dovolj preprosti, saj je aplikacija namenjena mobilnim napravam. Nekaterim modelom je bilo potrebno teksture spremeniti oziroma zmanjšati. Tematsko so usmerjeni v srednji vek. Model nasprotnika ima definirano tudi okostje tako,

[3] https://www.assetstore.unity3d.com/

Slika 6.2 3D modeli uporabljeni v sceni.

da je odprta možnost dodajanja novih animacij. Trenutno ima ta model dve animaciji. V prvi lik s sekiro seka drva, v drugi pa je lik nedejaven, pri čemer je nekoliko bolj intenzivno prikazano dihanje.

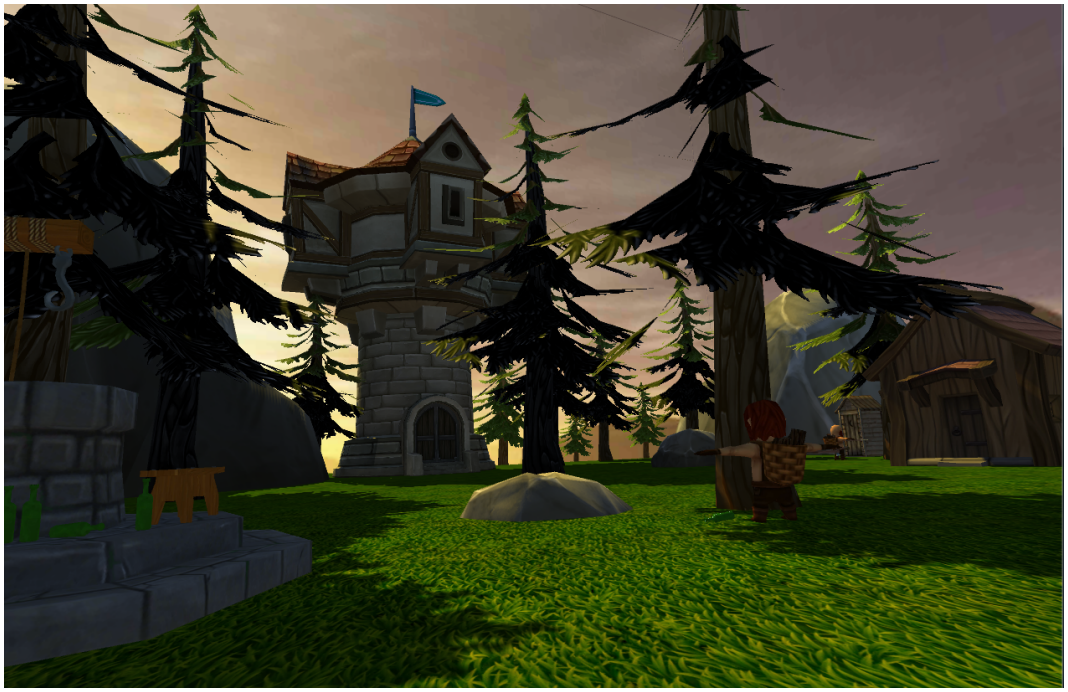
6.2 Aplikacija

Aplikacijo lahko razdelimo na dva dela. Prvi služi za vizualno predstavitev navideznega okolja (glej sliko 6.3), drugi pa za pridobivanje podatkov iz senzorjev naprave, ter razpoznavo kretenj. Sem spadajo tudi izpis podatkov in izvajanje meritev.

6.2.1 Navidezno okolje

Navidezno okolje predstavlja igralno polje razvite igre. Tema je srednjeveški gozd, z nekaj nasprotniki in ustreznimi stavbami, ki se s sceno tematsko ujemajo. Najprej sem v sceno vstavil hiše, pekarno in kamnit stolp. Te sem porazdelil po sceni in tematsko okrasil z vodnjakom, sodi, posodami in steklenicami.

Prav tako sem generiral hribovje in skale, ki popestrijo okolico z metanjem senc na pokrajino in zakrivanjem sonca iz določenih delov scene. Največje število objektov v



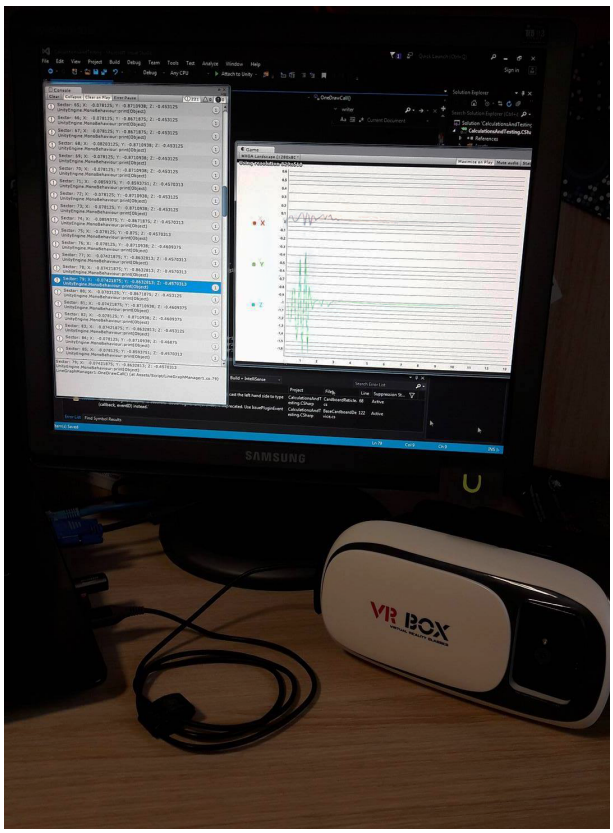
Slika 6.3 Zajet del scene igre.

sceni predstavlja množica dreves, ki tvorijo gozd. Ta so zgoščeno porazdeljena po mapi, med njimi pa so poti, ki vodijo do vseh središč dogajanja v igri.

6.2.2 Meritve, izpis in algoritem

V drugen delu aplikacije sem beležil izhodne podatke senzorjev mobilne naprave in analiziral ugotovitve. Uporabil sem podatke iz mobilnega merilca pospeška. Na koncu razvoja skripte za vse meritve in izračune pa sem izdelal še grafe, ki v realnem času, med izvajanjem različnih kretenj, prikazujejo podatke pridobljene iz senzorjev.

Naprej sem pripravil prikaz podatkov senzorja mobilne naprave na zaslon. Mobilna naprava, ki se je nahajala v VR očalih, je bila na računalnik povezana preko USB povezave. Na mobilni napravi se je izvajal program Unity Remote 4, ki omogoča istočasno povezavo in prenos programa iz okolja Unity 3D v napravo in obratno. Prav tako omogoča prenos podatkov iz mobilne naprave v okolje Unity 3D, kjer se ti lahko uporabijo. V okolju Unity 3D sem v programskem jeziku C# napisal program, ki iz pridobljenih podatkov izrisuje graf v realnem času. Merilec pospeška mobilne naprave meri pospeške na treh koordinatnih oseh (X, Y in Z). Te sem ločeno izpisoval na isti graf in jih ustrezno



Slika 6.4 Pridobivanje in izrisovanje podatkov na graf iz mobilne naprave, povezane z okoljem Unity 3D.

obarval. Vrednosti vseh treh pospeškov se ob normalnem gibanju uporabnika zadržujejo znotraj mej $\pm$ 1, a ob sunkovitem gibu, sile dosežejo vrednosti do nekje $\pm$ 2. Graf mi je pomagal pri približni vizualni oceni dogajanja medtem, ko so mi neobdelani podatki v konzoli podali natančno povprečno vrednost, kar sem kasneje potreboval pri izdelavi programa za interakcijo z okoljem. Surove podatke je program ob vseh meritvah urejeno shranjeval v tekstovno datoteko. Slika 6.4 prikazuje potek pridobivanja podatkov iz mobilne naprave in njihov izris na graf.

7 Meritve in implementacija kontrol v igro

Skozi celoten proces je bilo potrebno določiti sekvenco zaznavanja podatkov. Po različnih testiranjih aplikacije na mobilnih napravah, kot so odzivnost aplikacije, segrevanje naprav in uspešnost algoritma pri prepoznavanju podatkov, sem ugotovil, da je 10 zajemov na sekundo najbolj ustrezna sekvenca zajemanja podatkov.

Pri zajemanju, s sekvenco 100 podatkov na sekundo in več se mobilnim napravam v kratkem času drastično poveča delovna temperatura. To je še toliko bolj opazno pri napravah, ki ne sodijo v najvišji cenovni razred in posledično niso najbolj zmogljive.

V primeru, ko se sekvenca zajemanja podatkov zmanjša pod 8 zajemov na sekundo, se napravam delovna temperatura sicer ne povečuje, a se zelo poslabša odzivnost aplikacije. Posledično se pokvari tudi uporabniška izkušnja.

Za potrebe meritev sem v okolju Unity 3D spisal knjižnico, ki na zaslon računalnika v realnem času na graf izrisuje zajete vrednosti. Graf prikazuje vrednosti sil po koordinatnih oseh X, Y in Z merilca pospeška (glej sliko 5.1) v odvisnosti časa.

Os X merilca pospeška zaznava pospeške, ki jih ustvarja gibanje glave v levo oziroma v desno. Obarvana je z rdečo barvo. Lega mobilne naprave v VR očalih je horizontalna

in z zaslonom obrnjena proti uporabniku. V začetni poziciji mobilne naprave ima senzor koordinate osi X vrednost 0. Os Y merilca pospeška zaznava sile, ki se izvajajo v vertikali glede na trenutno pozicijo mobilne naprave. V začetnem stanju je njena vrednost -1. Barva za njen prikaz v grafu je zelena. Os Z merilca pospeška pa zaznava sile sile v globino. To so uporabnikovi gibi naprej in nazaj glede na začetno stanje. V grafu je obarvana z modro barvo in njena začetna vrednost je 0. Pri meritvah je potrebno delno upoštevati tudi to, da so lahko VR očala različnih proizvajalcev, različnih materialov, tež in kvalitet. S tem pride do delnih odstopanj pri podatkih, ki jih vrača senzor. Naprave ob premiku uporabnika prav tako različno med seboj nihajo na njegovi glavi.

Meritve sem izvajal z mobilno napravo, priklopljeno preko USB kabla na osebni računalnik. Za vsak poizkus sem zajemal vrednosti vseh treh koordinatnih osi merilca pospeška v odvisnosti od časa. Vsako sekundo sem zajel 10 vrednosti in te izrisal na graf. Zajemal sem 13 sekund in tako izrisal 130 prebranih vrednosti. Istočasno pa so se surovi podatki shranjevali v podatkovno datoteko, iz katere sem kasneje natančneje razbral in izračunal izmerjene podatke.

V eksperimentih namenjenih razvoju algoritma je sodelovalo deset oseb. Da bi bili podatki še bolj nevtralni, so bile osebe obeh spolov in različnih starostnih razredov. Sodelovalo je pet moških in pet žensk, od 22 let starosti pa vse do 57 let starosti. Nekaj od oseb je že preizkusilo navidezno resničnost in jim ta izkušnja ni bila tuja, nekaj jih je zanjo samo slišalo, dvema pa je bila navidezna resničnost tuja. Vsi so opravili 10 ponovitev za vsako izmed želenih interakcij in za vsako intenziteto le-te. Grafe in zajete podatke sem kasneje pregledal in s pomočjo podatkovnih datotek izračunal povprečne mejne vrednosti, potrebne za razpoznavanje določene interakcije. Vse vrednosti mejnih točk, ki se uporabljajo za razpoznavanje interakcij so bile izračunane na osnovi podatkov iz tekstovnih datotek. Grafi so služili le kot vizualni prikaz zajetih vrednosti. Z njimi sem si pomagal pri razvrščanju iz tekstovnih datotek. Zelo so namreč pripomogli pri orientaciji po tekstovnih datotekah. Tako sem lažje našel vrhove, mejne vrednosti za algoritem, ki prepoznava, katera interakcija se izvaja. Vsem tem vrhovom sem izračunal povprečne vrednosti in opazoval, kako se glede na uporabnika pri določenih kretnjah spreminjajo.

Ko je bil algoritem končan sem izdelal tudi grafe za vse tri tipe interakcije. Ti vizualno prikazujejo razliko sil med tremi različnimi osebami, udeleženimi v testiranjih. Prva oseba je moškega spola, stara 27 let in je dobro seznanjena z navidezno resničnostjo in

aplikacijami, ki jih ta nudi. Druga oseba je ženskega spola, stara 25 let in sicer navidezno resničnost pozna a je še ni preizkusila. Tretja oseba je ženskega spola, stara 52 let in ji je navidezna resničnost in njena tehnologija neznana. Ugotovili smo, da testiranci, ki jim navidezna resničnost ni tuja, prej osvojijo tehniko gibov kar vodi k uspešnejši prepoznavi kretenj [17].

7.1 Hoja

Najprej sem razvil zaznavanje hoje. Tu lahko horizontalne sile z X in Z koordinatnih osi zanemarimo, saj je mišljeno, da uporabnik hodi na mestu. Na grafu (glej sliko 7.1) lahko opazimo, da se njune vrednosti rahlo spreminjajo, a to je zaradi motenj pri tresljajih na očala. Zanimajo nas sile s koordinatne osi Y merilca pospeška. Izvedli smo več načinov in intenzivnosti hoje: visoko intenzivno hojo, srednjo intenzivno in sproščeno hojo.

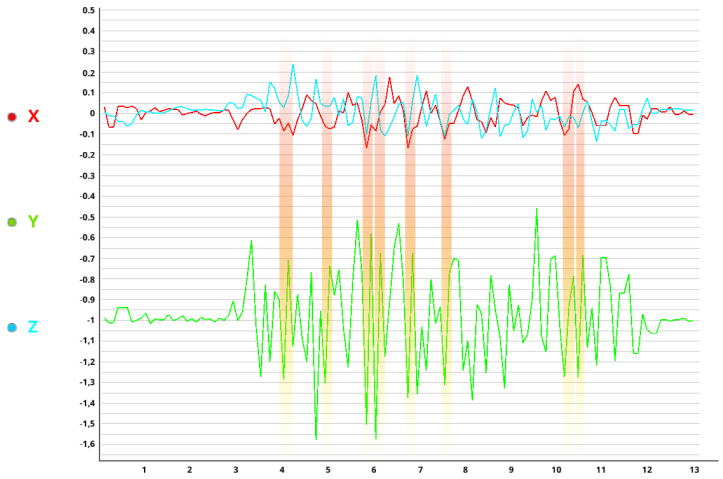
Sproščena intenzivnost hoje ima premalo specifične sile in hitro pride do napačnih zaznav hoje. Že majhen tresljaj mobilne naprave algoritem zazna kot hojo. Pri visoki intenzivnosi hoje pa pride do preveč različnih podatkov med uporabniki, saj eni hojo izvajajo bolj, drugi manj intenzivno. Meje zaznavanj se prav tako približajo mejnim silam zaznave pri skoku.

Intenzivnost hoje, ki sem jo izbral in je prikazana na grafu (glej sliko 7.1), je srednja intenzivnost hoje. Razvidno je, da so podatki sorazmeroma ciklični in doseženi vrhovi/dna konstantni. Povprečna izmerjena vrednost maksimalnih/minimalnih sil na koordinatni osi Y ob koraku je -1.28 pri udarcu ob tla in -0,79 ob odrivu nazaj v zrak. Vrednosti so negativne v obeh primerih, ker se vrednost sil koordinatne osi Y v mirujočem položaju nahajajo okoli -1.

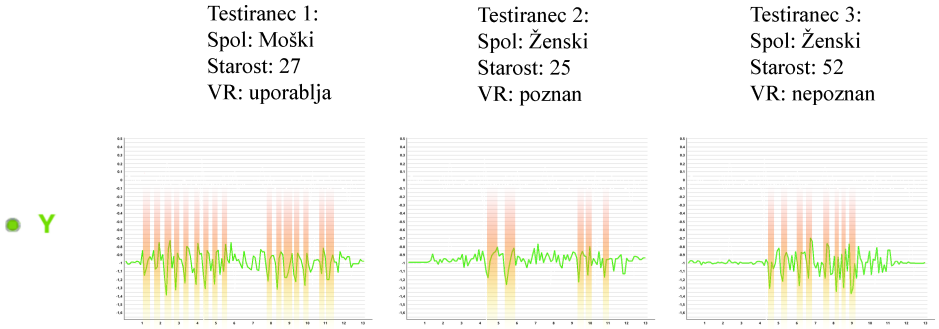
Slika 7.2 prikazu grafe treh različnih testirancev. Na njih so prikazane le vrednosti koordinatne osi Y merilca pospeška. Na sliki so z rdeče-oranžnimi površinami označena polja, kjer testiranec izvaja kretnjo.

7.2 Skok

Kot pri hoji, lahko tudi pri skoku zanemarimo podatke koordinatnih osi X in Z, saj so pri skoku nepomembni. Pomembni bi bili, če bi implementirali skok v različne smeri. Merili smo različne intenzivnosti skokov in izbral sem blažjo obliko. Skok je tolikšen, da uporabnik komaj da odskoči od tal. Na grafu so ti poskoki uspešno razvidni iz zajetih



Slika 7.1 Graf reprezentativne meritve pospeškov, ki nastanejo pri hoji. Označeni sektorji predstavljajo uspešno prepoznavo kretnje.



Slika 7.2 Grafi pospeškov pri hoji treh različnih testirancev. Označeni sektorji predstavljajo izvajanje kretnje.

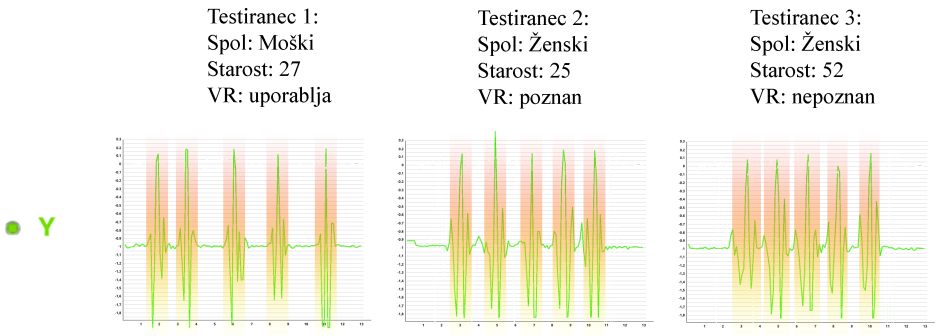
Slika 7.3 Graf reprezentativne meritve pospeškov, ki nastanejo pri skoku. Označeni sektorji predstavljajo uspešno prepoznavo kretnje.

vrednosti sil na Y osi. Različno kot pri grafu hoje, so tu podatki vidno bolj ločeni v skupine. V tem primeru smo izmerili 5 poskokov. Ti so si med seboj podobni, kar je lepo razvidno tudi na grafu (glej sliko 7.3). Minimalna vrednost spodnje zaznave je -1.79, zgornje pa -0.45. Tu pa bo zaznavanje nekoliko drugačno kot pri hoji. V primeru zaznave negativne vertikalne sile, z vrednosti z vsaj -1.79 (nabiranje uporabnikovega odriva), se bo program začel zavedati, da morda ne gre le za hojo, ampak morda za skok. Nato meri naslednjih nekaj vrednosti sil. Za tem mora slediti vrednost vertikalne sile višje od -0.45. Ta predstavlja zaznano silo na koordinatni osi Y pri odskoku. Zaznava se lahko izvaja še naslednja 2 časovna koraka, v primeru, da je odskok uporabnika morda nekoliko zakasnjen. Pristanek ter vrnitev v stoječo pozicijo je zanemarjena, saj bi se zaradi tega poslabšala igralnost in odzivnost igre.

Slika 7.4 prikazu grafe treh različnih testirancev. Na njih so prikazane le vrednosti koordinatne osi Y merilca pospeška. Na sliki so z rdeče-oranžnimi površinami označena polja, kjer testiranec izvaja kretnjo.

7.3 Met

V aplikacijo sem vključil tudi zaznavanje meta. Pri metu je z začetno kalibracijo potrebno natančneje nastaviti mejne vrednosti zaznave za vsakega uporabnika posebej. Kot je raz-



Slika 7.4 Grafi pospeškov pri skoku treh različnih testirancev. Označeni sektorji predstavljajo izvajanje kretnje.

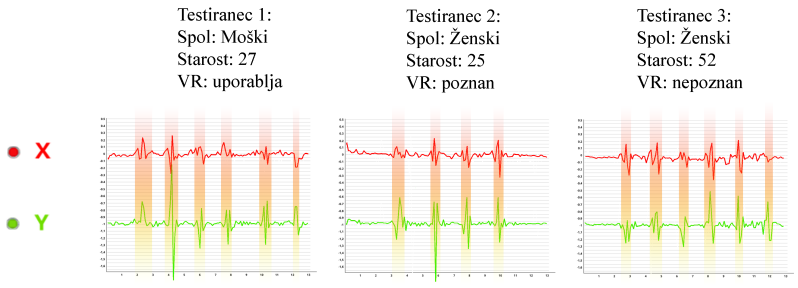
vidno iz grafa, se pri meritvah meta pojavljajo spremembe na vseh treh koordinatnih oseh merilca pospeška (glej sliko 7.5). Sile na koordinatni osi Z so se med opazovanci in posameznimi poskusi preveč razlikovale, zato sem jo pri zaznavanju izpustil. Za prepoznavo kretnje giba met algoritem uporablja zgolj koordinatni osi X in Y. Ob metu se vse sile, ki jih merilec pospeška zazna odvijejo v zelo kratkem časovnem obdobju. Prav tako, kot pri prejšnjih dveh kretnjah, sem merili tri intenzitete meta. Srednja intenziteta se je izkazala kot najbolj ustrezna. Ta ima izmed vseh intenzitet pri opazovancih najbolj podobne sile. Algoritem zaznava 3 korake (vzorce) oziroma 0,3s. Ob zaznavi prvega prepoznanega giba kretnje meta, se ta zabeleži in vklopi poslušalec (angl. listener), ki traja 5 časovnih korakov (0,5 sekunde). V tem času se morajo zaznati naslednja tri mejna območja. Ko uporabnik izvede kretnjo meta, se v ramenskem delu nekoliko zavrti proti dominantni roki, ki met izvaja. To predstavlja zalet, ki ga uporabnik izvede, da bo lahko stvar zalučal. Ta gib algoritem zanemari, saj se ne pojavi dovolj intenzivno pri vseh testirancih. Sledi izteg roke, ki predstavlja met. Povprečna vrednost po koordinatni osi X mora za ustrezno delovanje preseči 0.07. Po uspešni zaznavi se mora v roku 2 časovnih korakov zaznati še pospešek, ki na koordinati osi Y preseže mejo -0.83. Šele, ko sta obe sili ustrezno zaznani, sledi drugi korak zaznavanja. Ta je omah uporabnikove roke. Tu mora povprečna sila v roku dveh časovnih korakov na koordinatni osi X pasti pod -0.16, istočasno v teh dveh časovnih korakih pa mora biti zaznan padec sile na koordinatni osi Y pod mejo -1.18. Le v primeru, ko so vsi pogoji ustrezno zaznani, algoritem izvede interakcijo met. Zaznavanje te kretnje je tako nekoliko kompleksnejše kot hoja ter skok,

Slika 7.5 Graf reprezentativne meritve pospeškov, ki nastanejo pri metu. Označeni sektorji predstavljajo uspešno prepoznavo kretnje.

zato se mora uporabnik nanj tudi nekoliko privaditi.

Omenjene sile za zaznavo meta veljajo za uporabnike, ki so desničarji in izvajajo kretnjo z desno roko. V primeru, da je uporabnik levičar, se spremeni le zaznavanje sil koordinate X, pri kateri se zahtevana vrednost zrcali preko začetne meje, na kateri se sile na tej koordinati nahajajo v začetnem kalibriranem stanju.

Slika 7.6 prikazu grafe treh različnih testirancev. Na njih so prikazane le vrednosti koordinatnih osi X in Y merilca pospeška. Na sliki so z rdeče-oranžnimi površinami označena polja, kjer testiranec izvaja kretnjo.



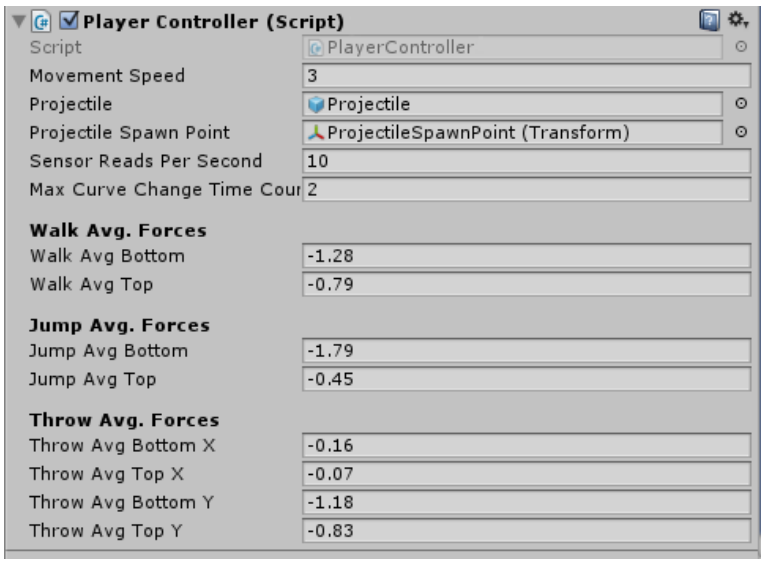
Slika 7.6 Grafi pospeškov pri metu treh različnih testirancev. Označeni sektorji predstavljajo izvajanje kretnje.

8 Program

Algoritem sem napisal v programskem jeziku C#. Nahaja se v eni datoteki. Temelji na znavanju podatkov iz merilca pospeška mobilne naprave skozi čas. Trenutno je implementirana zaznava 3 kretenj: hoja, skok ter met. Vsi parametri so razvijalcu dostopni v kontrolnem oknu (angl. inspector) objekta, na katerem se program nahaja (glej sliko 8.1). Tu lahko pred zagonom kalibrira povprečne mejne vrednosti zaznavanja in jih prilagodi na uporabnika. Algoritem je razdeljen v 2 fazi.

8.1 Prva faza

Prva faza skrbi za zaznavo hoje in skoka. Njuno zaznavanje je združeno v isto fazo algoritma, saj se s tem zmanjša nepotrebno dodajanje spremenljivk. Program ves čas v vnaprej nastavljenih časovnih korakih prejema podatke iz merilca pospeška. V prvi fazi se uporabljajo samo podatki koordinatne osi Y. V primeru, da algoritem zazna pospešek, nižji od minimalne spodnje povprečne vrednosti pri hoji, vstopi v pogoj prve faze. Tu si program zabeleži, da je bila zaznana sila, ki lahko predstavlja hojo. Prav tako se preveri in shrani, če je bila sila nižja od spodnje povprečne vrednosti pri skoku. Program ima



Slika 8.1 Zgled kontrolnega okna (angl. inspector) skripte algoritma.

sedaj podatek, ali je bila zaznana sila, ki predstavlja prvo fazo kretnje hoje in podatek, če gre morda za začetno fazo kretnje skoka. Če začetna zaznana sila ni bila nižja od povprečne spodnje meja skoka, se algoritem izvaja naprej. Za uspešno potrditev hoje v naslednjem časovnem koraku pričakuje poskok sil preko zgornje absolutne meje hoje. V obratnem primeru, ko je začetna zaznana sila nižja od povprečne spodnje meje skoka, pa se zaznavanje hoje zakasni za tri časovne korake (0,3s) in v tem času algoritem čaka na ustrezne sile druge faze kretnje skoka. V prvem časovnem koraku algoritem zaznava sile in si zapomni, ali je prišlo do pravilne zaznave prekoračitve sil preko zgornje absolutne meje pri hoji. Ta podatek uporabi za potrditev hoje v primeru, če se v naslednjih dveh časovnih korakih ni uspešno zaznala prekoračitev zgornjih absolutnih mej skoka. Če pa je v enem izmed treh časovnih korakih uspešno zaznan poskok sil nad zgornjo absolutno vrednost kretnje skoka, pa se ta uspešno zazna in izvede. Ob izvedbi skoka se nastavi spremenljivka, da je igralec v zraku. Ta se ponastavi šele ob pristanku na tla. Med časom, ko je ta nastavljena, se prva faza algoritma ne izvaja. Na koncu uspešne oz. neuspešne zaznave katere koli kretnje se pogojne spremenljivke ponastavijo na začetne vrednosti. Prvo fazo programa, ki zaznava hojo in skok, prikazuje izpis 8.1.

Izpis 8.1 Prva faza algoritma kateri zaznava hojo in skok.

```
void MovementAndJumpRecognition() {
        if (jumpFailedWalkTrigger && (jumpTimer == 0)) {
                Walk();
                ResetWalkJumpVariables();
        } else if (walkFirstPhaseDetected) {
                if(!jumpFirstPhaseDetected && (Input.acceleration.y > walkAvgTop)) {
                        Walk();
                        ResetWalkJumpVariables();
                }
                if (walkFirstPhaseDetected) {
                        if (walkTimeCounter > 2) {
                                walkTimeCounter = 0;
                                walkFirstPhaseDetected = false;
                        } else walkTimeCounter++;
                }
                if (jumpFirstPhaseDetected) {
                        if(jumpTimer == 3 && (Input.acceleration.y > walkAvgTop))
                             jumpFailedWalkTrigger = true;
                        CheckForJumpSeccondStep();
                        jumpTimer--;
                }
                if(jumpFirstPhaseDetected && jumpTimer == 0) {
                        if (jumpFailedWalkTrigger) {
                                ResetWalkJumpVariables();
                                jumpFailedWalkTrigger = true;
                        } else ResetWalkJumpVariables();
                }
        }
        else if (grounded) {
                if(Input.acceleration.y < walkAvgBottom) {
                        walkFirstPhaseDetected = true;
                                if (Input.acceleration.y < jumpAvgBottom) {
                                jumpFirstPhaseDetected = true;
                                jumpTimer = 3;
                        }
                }
        }
}

void ResetWalkJumpVariables() {
        walkFirstPhaseDetected = false;
        walkTimeCounter = 0;
        jumpFirstPhaseDetected = false;
        jumpFailedWalkTrigger = false;
        jumpTimer = 0;
}

void CheckForJumpSeccondStep(){
        if(Input.acceleration.y > jumpAvgTop){
                Jump();
                grounded = false;
                ResetWalkJumpVariables();
        }
}

void Walk(){
        transform.position += new Vector3(Camera.main.transform.forward.x * movementSpeed,
             0f, Camera.main.transform.forward.z * movementSpeed);
}

void Jump(){
        transform.Translate(Vector3.up * 5f, Space.World);
}
```

8.2 Druga faza

Druga faza algoritma skrbi za zaznavo meta uporabnika. V tej fazi se uporabljajo sile X in Y koordinatnih osi merilca pospeška mobilne naprave. Pri zaznavanju meta se od prve ustrezno zaznane kretnje uporablja odštevalnik. Vse potrebne zaznave se morajo zgoditi znotraj časa odštevalnika, drugače se spremenljivke, uporabljene za zaznavo meta, ponastavijo na začetne vrednosti. Pogoj za vstop v drugo fazo algoritma je pravilna zaznava začetnih sil kretnje meta. Zaznana mora biti sila X koordinate merilca pospeška, ki prekorači zgornjo absolutno mejo meta po horizontali (rotacija uporabnika levo in desno) in sila Y koordinate merilca pospeška, ki prekorači zgornjo absolutno mejo meta po vertikali (izmet). Zaradi pravilnosti zaznave in dopuščanju manjših napak se morata ta dva pogoja izpolniti v razmiku 2 časovnih korakov, za kar uporabljam kar časovnik, v nasprotnem primeru se vrednosti spremenljivk povrnejo na začetne vrednosti. Ko sta izpolnjena v razmaku dveh časovnih korakih, program na novo nastavi odštevalnik na pet časovnih korakov in vstopi v drugo fazo algoritma. Met se izvede, če sta v času odštevalnika zaznani sila X koordinate merilca pospeška, ki je nižja od spodnje absolutne meje meta po horizontali, in sila Y koordinate merilca pospeška, ki je nižja od spodnje absolutne meje meta po vertikali. Spodnje in zgornje absolutne meje se na začetku igre nekoliko kalibrira in prilagodi na uporabnika. Drugo fazo, ki zaznava met, prikazuje izpis 8.2.

Izpis 8.2 Druga faza algoritma kateri zaznava met.

```
void ThrowRecognition() {
        if (throwFirstPhaseDetected) {
                if (Input.acceleration.y < throwAvgBottomY) throwYdone = true;
                if (Input.acceleration.x < throwAvgBottomX) throwXdone = true;

                throwTimeCounter++;

                if (throwXdone && throwYdone) {
                        Throw();
                        throwFirstPhaseDetected = false;
                        throwXdone = false;
                        throwYdone = false;
                        throwTimeCounter = 0;
                }

                if (throwTimeCounter > 5){
                        throwFirstPhaseDetected = false;
                        throwXdone = false;
                        throwYdone = false;
                        throwTimeCounter = 0;
                }
        } else {
                if (Input.acceleration.x > throwAvgTopX && !throwXdone) {
                        throwXdone = true;
                        if (!throwYdone) throwTimeCounter = 0;
                }
                if (Input.acceleration.y > throwAvgTopY && !throwYdone) {
                        throwYdone = true;
                        if (!throwXdone) throwTimeCounter = 0;
                }

                throwTimeCounter++;

                if (throwTimeCounter > 2) {
                        throwXdone = false;
                        throwYdone = false;
                        throwTimeCounter = 0;
                }
        }

        if (throwXdone && throwYdone) {
                throwFirstPhaseDetected = true;
                throwXdone = false;
                throwYdone = false;
                throwTimeCounter = 0;
        }
}

void Throw(){
        GameObject bullet = (GameObject)Instantiate(projectile,
             projectileSpawnPoint.position, Quaternion.identity);
        bullet.GetComponent<Rigidbody>().AddForce(
        Camera.main.transform.forward * 500);
}
```

9 Ugotovitve

Navidezna resničnost nam je iz dneva v dan bolj na dosegu roke. Integracija le-te v mobilni svet njeno dosegljivost še toliko poveča. Prav tako se širi njena uporabnost. Ker je vse več mobilnih naprav dovolj zmogljivih, da podpira navidezno resničnost a mnogo očal za vizualizacijo navidezne resničnosti s pomočjo mobilnih naprav ne vsebuje nobenih upravljalnikov, je razvoj brezročne interakcije obvezen. V diplomski nalogi sem skozi meritve spoznal, da medtem ko senzorji lahko zaznavajo osnovne gibe, je ta prenos v aplikacijo še vedno le začasen način kontrol. Manjka boljša natančnost senzorjev. Slaba natančnost senzorjev se še posebej opazi pri starejših napravah.

S svojim algoritmom sem zaznaval osnovne načine interakcije, ki pridejo prav pri upravljanju z navideznimi svetovi. Tega sem prilagodil, da deluje na večini naprav z vgrajenim merilcem pospeška. Enostavna je tudi implementacija algoritma v projekt. Knjižnica je dokumentirana in enostavna za implementacijo. Potrebno jo je pritrditi na glavno kamero v sceni v okolju Unity 3D, kalibrirati mejne vrednosti za zaznavo gibov in aplikacijo prenesti na mobilno napravo. Seveda mora naprava imeti merilec pospeška in biti dovolj zmogljiva za izvajanje VR aplikacij.

S ciljem preveriti uporabnost takšnega načina interakcije sem izvedel tudi anketo. Anketiral sem 7 oseb, katerim sem pred anketo predstavil svoj program. Ankentiranci so ga videli prvič. Kot pri testirancih sem poskušal najti 7 raznolikih ankentirancev. Njihova leta se gibajo med 19 in 56 let. Razlikujejo se tudi v poznavanju navidezne resničnosti. Anketiral sem oba spola. Vprašanja, ki sem jih postavljal so sledeča:

- odzivnost hoje, skoka in meta (0 - 10),
- občutek naravnosti kretnje hoje, skoka in meta v igri (0 - 10).

Pred začetkom sem vsakemu anketirancu kalibriral njegove mejne vrednosti za zaznavanje kretenj. Igro so igrali deset minut. Anketo sem pri vsakem izvedel trikrat. Na začetku igre, po petih minutah igranja in na koncu igranja igre. Razvidno je bilo, da dlje kot so igrali igro in se privajali, boljši so bili rezultati ankete. Največ težav so imeli tisti ankentiranci, kateri navidezne resničnosti niso poznali. Ti pa so pokazali največjo spremembo v odgovorih na anketo skozi igro. Tabeli 9.1 in 9.2 prikazujeta povprečne odgovore vseh ankentiracnev s standardnim odklonom.

Tabela 9.1 Anketa odzivnosti kretenj. N=7.

	1 min.	5 min.	10min.
Hoja	4 ± 1,05	8 ± 0,73	9 ± 0,73
Skok	2 ± 2,14	5 ± 1,13	7 ± 1,12
Met	1 ± 0,73	3 ± 0,88	4 ± 1,18

Tabela 9.2 Anketa naravnosti kretenj. N=7.

	1 min.	5 min.	10min.
Hoja	2 ± 1,03	5 ± 0,73	6 ± 1,05
Skok	3 ± 0,62	5 ± 0,83	5 ± 0,7
Met	1 ± 0,25	2 ± 0,83	3 ± 1,5

10 Zaključek

Kljub temu, da so današnje mobilne naprave že pravi majhni računalniki, so senzorji še vedno nekoliko premalo razviti, da bi se ti uporabljali za interakcijo v navideznih okoljih s kretnjami. V diplomski nalogi sem ustvaril knjižnico, ki izrablja podatke prejete iz senzorjev v mobilni napravi za zaznavo osnovnih kretenj. Te so namenjene interakciji v navideznem okolju. Uporabniku še vedno manjka druga najbolj pomembna informacija pri predstavitvi navideznega okolja – povratna informacija. Te pa drugače kot z vizualnimi efekti na ekranu, zvočnimi efekti oziroma z uporabo vibracij v mobilni napravi ni mogoče ustvariti. Mnogo podjetij se ukvarja ravno s tem problemom. Pričakujemo lahko, da bo razvoj VR tehnologije za povratne informacije v bližnji prihodnosti razcvetel in da bo tehnologija dostopna tudi za mobilne naprave, kar pa bo zelo popestrilo doživetje mobilnih VR aplikacij.

Literatura

[1] T. Mazuryk, M. Gervautz, Virtual reality history, applications, technology and future, Human contact: technical-report (1996) 1–21.

[2] M. Justin, A. Sexton, The history of virtual reality (2016).
url: http://www.tomshardware.com/picturestory/704-history-of-virtual-reality.html

[3] S. Cobb, A. Moody, J. R.Willson, Virtual reality-induced symptoms and effects (vrise), Teleoperators and Virtual Environments (2007) 171–172.

[4] R. J. K. Jacok, What you look at is what you get: Eye movement-based interaction techniques, Proceedings of the SIGCHI Conference on Human Factors in Computing Systems Empowering People - CHI (1990) 11–18.

[5] Rift, Oculus rift vr motion sickness – 11 ways to prevent it (2016).
url: http://riftinfo.com/oculus-rift-motion-sickness-11-techniques-to-prevent-it

[6] B. G. Blundell, J. S. Adam, Creative 3-D Display and Interaction Interfaces: A Trans-disciplinary Approach, Hoboken (New Jersey): Wiley-Interscience, New Jersey, 2006.

[7] C. Christou, A. Tzanavari, K. Herakleous, P. Charalambos, Navigation in virtual reality: Comparison of gaze-directed and pointing motion control, Mediterranean Electrotechnical Conference (MELECON) (2016) 3–5.

[8] R. J. K. Jacok, The use of eye movements in human-computer interaction techniques: What you look at is what you get, ACM Transactions on Information Systems TOIS ACM Trans. Inf. Syst. (1991) 152–169.

[9] K. Goode, Stare to win – a bad trend in vr design - triangular pixels (2015). url: http://www.triangularpixels.net/cms/development/stare-to-win-a-bad-trend-in-vr-design/

[10] E. Rae A., R. M. A. Gigante, J. H., Virtual reality systems, London: Academic (1993) 93–118.

[11] O. Grau, Virtual art: From illusion to immersion, Cambridge, MA: MIT (2003) 11–23.

[12] S. Stanković, Virtual reality and virtual environments in 10 lectures, Synthesis Lectures on Image, Video, and Multimedia Processing (2015) 35–50.

[13] L. Terziman, M. Maud, E. Mathieu, F. Multon, B. Arnaldi, A. Lécuyer, Shake-your-head. revisiting walking-in-place for desktop virtual reality, Proceedings of the 17th ACM Symposium on Virtual Reality Software and Technology - VRST '10.

[14] J. Decety, M. Jeannerod, Mentally simulated movements in virtual reality: Does fitt's law hold in motor imagery?, Behavioural Brain Research (1995) 127–134.

[15] W. Herbert, J. Pfusterschmied, M. Klous, S. P. Von Duvillard, E. Müller, Movement variability and skill level of various throwing techniques, Human Movement Science (2012) 83–88.

[16] W. Lin-Hwa, L.-C. Kuo, S.-W. Shih, K.-C. Lo, K.-C. Lo, F.-C. Su, Comparison of dominant hand range of motion among throwing types in baseball pitchers, Human Movement Science (2013) 3–7.

[17] P. Thies, A. Schmidt, P. Renner, Detecting movement patterns from inertial data of a mobile head-mounted-display for navigation via walking-in-place, IEEE Virtual Reality (VR) (2016) 263–264.